# Presupuesto y Análisis Técnico para un Sistema de Gestión de Laboratorios con Arquitectura Escalable en la Nube

Alexandra Tinjacá Cortés, Juan Pablo Camacho Peñata, Miguel Ardila, Daniel Santiago Parra Escobar
Facultad de Ingeniería, Programa de Ingeniería de Sistemas
Universidad Piloto de Colombia
Bogotá, Colombia

***Resumen*—Este documento presenta un presupuesto detallado y un análisis técnico para el desarrollo, despliegue y mantenimiento de un sistema de gestión de reservas y préstamos de laboratorio. Se incluye una comparativa exhaustiva de plataformas en la nube, un análisis de rendimiento basado en transacciones por segundo (TPS), costos de infraestructura y salarios para un equipo de desarrolladores junior. La arquitectura propuesta —desplegada completamente en AWS utilizando EC2 para frontend/back y RDS para la base de datos— garantiza escalabilidad, bajo costo inicial y un camino de crecimiento claro y económico.**

***Index Terms*—Base de datos relacional, API REST, computación en la nube, TPS, presupuesto de TI, escalabilidad, AWS.**

## I. Introducción

La implementación de un sistema software profesional requiere una planificación técnica y financiera rigurosa. Este documento extiende su propuesta mediante un análisis detallado de costos, plataformas y rendimiento. Se consideran escenarios realistas de uso, transacciones por segundo (TPS) y salarios de mercado para desarrolladores junior en Colombia. El objetivo es proporcionar un presupuesto transparente y reproducible que sirva como base para la implementación real del sistema.

## II. Metodología

Los costos se calcularon en pesos colombianos (COP) utilizando una tasa de cambio de $4000 COP/USD. Los salarios de los desarrolladores junior se basaron en promedios de mercado para Colombia [12]. Los precios de las plataformas en la nube se obtuvieron de sus sitios web oficiales en septiembre de 2023. El análisis de TPS se realizó considerando un escenario de uso moderado-alto en un entorno universitario.

## III. Análisis de Rendimiento: Transacciones Por Segundo (TPS)

El rendimiento del sistema se midió en términos de transacciones por segundo (TPS), una métrica crítica para sistemas transaccionales basados en web [13].

**Parámetros de diseño:**

- Población total del sistema: 100 usuarios
- Porcentaje de concurrencia: 15 %
- Usuarios concurrentes: $100 \times 0{,}15 = 15$
- Transacciones por usuario por segundo: 2 (navegación normal + acción)

**Cálculo de TPS:**

$$\begin{aligned} \text{TPS} &= \text{Usuarios} \times \text{Transacciones} \\ &= 15 \times 2 = 30 \text{ TPS} \end{aligned}$$

Para manejar la carga estimada de 30 TPS se contempla en el presupuesto las opciones de pago para las fases de prueba y despliegue, se requerirá:

- RDS PostgreSQL db.t3.micro o db.t3.small (según carga) — soporta +500 TPS
- EC2 t3.micro/t3.small para backend y frontend — soporta cargas web de 150 TPS con autoescalado
- Implementación de técnicas de optimización: caching, compresión y consultas optimizadas

## IV. Comparativa de Plataformas y Selección Técnica

Para la capa de persistencia, se evaluaron cuatro opciones principales. Supabase [1] ofrece PostgreSQL administrado con autenticación integrada, API REST automáticas y políticas de Row Level Security (RLS). Firebase de Google [2] proporciona Firestore, una base de datos NoSQL orientada a eventos en tiempo real, aunque menos adecuada para consultas relacionales. PlanetScale [3] es una plataforma MySQL basada en Vitess que permite branching y escalamiento horizontal. AWS RDS [4] ofrece un servicio empresarial completamente gestionado, compatible con PostgreSQL, MySQL y otros motores, con capacidades de Multi-AZ, backups automáticos y escalamiento vertical. Debido a su integración nativa con la nube de AWS y su estabilidad para cargas productivas, RDS se considera la solución óptima para entornos con requerimientos de fiabilidad elevados.

Para el backend, se compararon Render [5], Heroku [6], Railway [7] y AWS EC2/Elastic Beanstalk [8]. Render destaca por su simplicidad y despliegues automáticos. Heroku ha reducido su oferta gratuita. Railway ofrece flexibilidad y modelo de pago por uso. AWS EC2, por su parte, permite control total sobre la infraestructura, seguridad granular mediante IAM, redes privadas (VPC) y autoescalado. Debido a la necesidad de integrar backend, base de datos y redes internas en un mismo

entorno seguro, EC2 resulta la opción más adecuada para un sistema institucional.

Para el frontend, se consideraron Netlify [9], Vercel [10] y GitHub Pages [11]. Aunque Netlify y Vercel ofrecen características avanzadas, la solución seleccionada fue AWS S3 junto con CloudFront, dado que permite almacenamiento de archivos estáticos a bajo costo, un CDN global de alto rendimiento y configuración unificada dentro del ecosistema AWS.

### *IV-A. Base de Datos*

Cuadro I
COMPARATIVA DE PLATAFORMAS DE BASE DE DATOS

| Plataforma | Plan Gratuito | Plan Básico (USD/mes) | Notas |
|---|---|---|---|
| Supabase | 2 DB, 8 GB, 5 GB BW | $25 | API REST y Auth integrados |
| Firebase | 1 GB, 50K lect./día | Pago por uso | No-SQL, en tiempo real |
| PlanetScale | 1 DB, 10 GB | $29 | MySQL, branching |
| AWS RDS | 750 hrs/mensuales (12 meses) | $13–$15 | PostgreSQL, Multi-AZ opcional |

**Selección: AWS RDS**. Ofrece alta disponibilidad, respaldo automático, seguridad empresarial y escalabilidad vertical inmediata.

### *IV-B. Servicio de Backend*

Cuadro II
COMPARATIVA DE PLATAFORMAS DE BACKEND

| Plataforma | Plan Gratuito | Plan Básico (USD/mes) | Notas |
|---|---|---|---|
| Render | 500 hrs/mes | $7 | Fácil uso, Git-integrado |
| Heroku | No gratis | $7 | Popular, add-ons |
| Railway | $5 crédito/mes | Pago por uso | Flexible |
| AWS EC2 | 750 hrs/mes (12 meses) | $9–$11 | Seguridad, VPC, autoescalado |

**Selección: AWS EC2**. Control total de red, seguridad avanzada y despliegue unificado con la base de datos.

### *IV-C. Servicio de Frontend*

**Selección: AWS S3 + CloudFront**. Bajo costo, alta velocidad y sin límites de framework.

### *IV-D. Integración de Plataformas Elegidas*

La arquitectura del sistema se compone de tres capas principales: presentación, aplicación y persistencia. La capa de persistencia se implementará utilizando AWS RDS con PostgreSQL, que ofrece respaldos automáticos, monitoreo, cifrado en reposo, redes privadas (VPC) y escalamiento vertical. La comunicación entre el backend y la base de datos se realizará mediante subredes privadas, evitando exposición pública.

Cuadro III
COMPARATIVA DE PLATAFORMAS DE FRONTEND

| Plataforma | Plan Gratuito | Plan Básico (USD/mes) | Notas |
|---|---|---|---|
| Netlify | 100 GB BW | $19 | CDN, serverless |
| Vercel | 100 GB BW | $20 | Optimizado para Next.js |
| GitHub Pages | 100 GB/mes | Gratis | Estático |
| AWS S3 + CloudFront | 5GB S3 + 1TB CF (free tier) | $1–$3 | CDN global, muy bajo costo |

La capa de aplicación se implementará en AWS EC2. El servidor backend (Node.js, Python, etc.) se ejecutará en una instancia t3.micro o t3.small, con la posibilidad de habilitar autoescalado y balanceo mediante un Application Load Balancer. Las políticas de seguridad se manejarán mediante AWS IAM y Security Groups, asegurando un entorno cerrado y auditable.

La capa de presentación estará alojada en AWS S3, donde se almacenará la aplicación web estática construida con el framework elegido. Su distribución se realizará mediante AWS CloudFront, un CDN global que reduce latencia y mejora el rendimiento. La comunicación entre el frontend y el backend se efectuará mediante HTTPS, utilizando certificados de AWS ACM integrados con CloudFront. Esta arquitectura unificada reduce costos operativos, garantiza un mayor control de seguridad y facilita escalar todas las capas del sistema dentro de un mismo entorno.

## V. PRESUPUESTO DETALLADO

### *V-A. Costos de Infraestructura*

Cuadro IV
COSTOS MENSUALES DE INFRAESTRUCTURA

| Item | Frecuencia | Costo Mensual (COP) | Total 3 meses (COP) |
|---|---|---|---|
| RDS PostgreSQL (db.t3.micro) | Mensual | $48,000 | $144,000 |
| EC2 t3.micro backend | Mensual | $41,000 | $123,000 |
| S3 + CloudFront (sitio estático) | Mensual | $12,000 | $36,000 |
| Dominio (.tech) | Anual | $10,700 | $32,000 |
| **Subtotal** | | **$111,700** | **$335,000** |

### *V-B. Costos de Recursos Humanos*

(Se mantiene igual)

### *V-C. Resumen General de Costos*

## VI. CONCLUSIÓN

La arquitectura propuesta —basada en AWS RDS, EC2 y S3+CloudFront— satisface los requisitos técnicos del sistema, proporcionando seguridad, escalabilidad y costos operativos bajos. El presupuesto total de tres meses se reduce significativamente gracias al uso del nivel gratuito de AWS, manteniendo una infraestructura robusta y apta para entornos institucionales.

Cuadro V
RESUMEN DE COSTOS TOTALES (3 MESES)

| **Concepto** | **Total COP (3 meses)** |
|---|---|
| Infraestructura | $335,000 |
| Recursos Humanos | $30,000,000 |
| **Subtotal** | **$30,335,000** |
| Fondo de contingencias (5 %) | $1,516,750 |
| **Total General** | **$31,851,750** |

## REFERENCIAS


[1] Supabase. “Supabase Documentation”. 2023. [En línea]. Disponible: https://supabase.com/docs

[2] Google. “Firebase Documentation”. 2023. [En línea]. Disponible: https://firebase.google.com/docs

[3] PlanetScale. “PlanetScale Documentation”. 2023. [En línea]. Disponible: https://planetscale.com/docs

[4] Amazon Web Services. “Amazon RDS Documentation”. 2023. [En línea]. Disponible: https://aws.amazon.com/rds/

[5] Render. “Render Documentation”. 2023. [En línea]. Disponible: https://render.com/docs

[6] Heroku. “Heroku Documentation”. 2023. [En línea]. Disponible: https://devcenter.heroku.com/

[7] Railway. “Railway Documentation”. 2023. [En línea]. Disponible: https://docs.railway.app/

[8] Amazon Web Services. “AWS Elastic Beanstalk Documentation”. 2023. [En línea]. Disponible: https://aws.amazon.com/elasticbeanstalk/

[9] Netlify. “Netlify Documentation”. 2023. [En línea]. Disponible: https://docs.netlify.com/

[10] Vercel. “Vercel Documentation”. 2023. [En línea]. Disponible: https://vercel.com/docs

[11] GitHub. “GitHub Pages Documentation”. 2023. [En línea]. Disponible: https://docs.github.com/pages

[12] “Estudio de Salarios para Desarrolladores Junior en Colombia,” Compensalab, 2023. [En línea]. Disponible: https://compensalab.com

[13] L. Richardson, M. Amundsen, S. Ruby, “RESTful Web APIs,” O’Reilly Media, 2013.

[14] “Supabase Pricing,” Supabase, Inc., 2023. [En línea]. Disponible: https://supabase.com/pricing

[15] “Render Pricing,” Render, Inc., 2023. [En línea]. Disponible: https://render.com/pricing

[16] “Netlify Pricing,” Netlify, Inc., 2023. [En línea]. Disponible: https://www.netlify.com/pricing